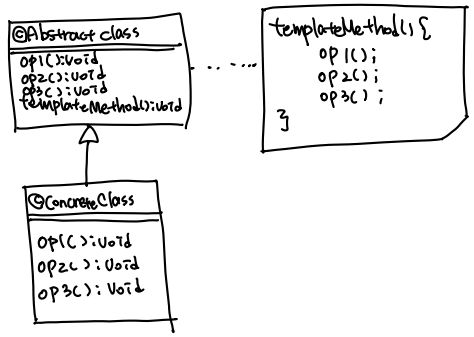
Template Method Pattern

2019.02.12

Abstract class

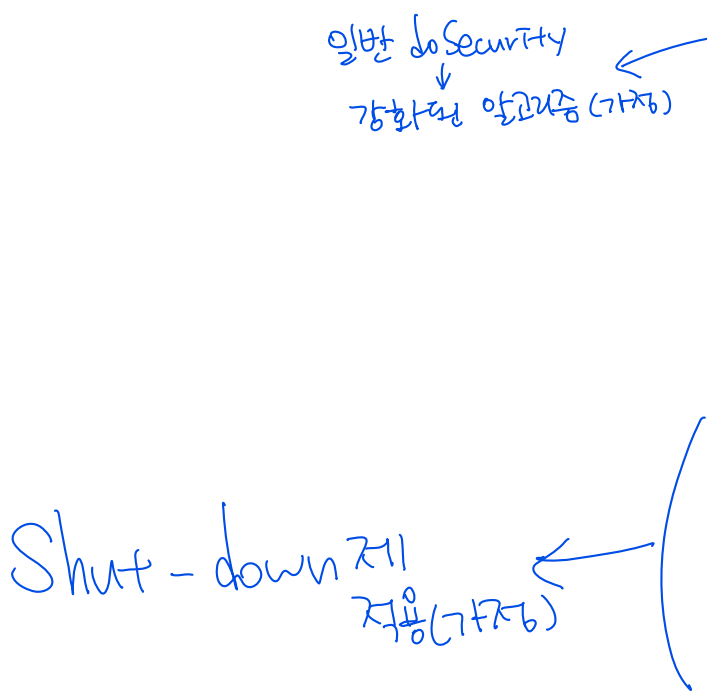
Steps in TemplateMethod

- Divide into Several stages.
- Declare the steps as a method.
- Create a template method.
- Implement the split methods




```java
package TemplateMethod;

public abstract class GameConnectHelper {
    protected abstract String doSecurity(String string);
    protected abstract boolean authentication(String id, String password);
    protected abstract int authorization(String userName);
    protected abstract String connection(String info);

    //템플릿 메소드
    public String requestConnection(String encodedInfo){

        //보안 작업 -> 암호화 된 문자열을 복호화
        String decodedInfo = doSecurity(encodedInfo);

        //반환된 것을 가지고 아이디, 암호를 할당한다.
        String id = "aaa";
        String password = "bbb";

        if(!authentication(id, password)){
            throw new Error("아이디, 암호 불일치");
        }

        String userName = "userName";
        int i = authorization(userName);

        switch(i){
            case -1:
                throw new Error("셧다운");
            case 0://게임매니저
                break;
            case 1://유료회원
                break;
            case 2://무료회원
                break;
            case 3://권한없음
                break;
            default:
                break;
        }

        return connection(decodedInfo);
    }
}
```

Class Implement




```java
package TemplateMethod;

public class DefaultGameConnectHelper extends GameConnectHelper {
    @Override
    protected String doSecurity(String string) {
        System.out.println("강화된 알고리즘을 이용한 디코드");
        return null;
    }

    @Override
    protected boolean authentication(String id, String password) {
        System.out.println("아이디/암호 확인 과정");
        return true;
    }

    @Override
    protected int authorization(String userName) {
        System.out.println("권한 확인");
        //서버에서 유저이름 유저의 나이를 알 수 있다.
        //나이를 확인하고 시간을 확인하고 성인이 아니고 10시가 지났다면
        //shut-down(if loop Algorithm)
        return 0;
    }

    @Override
    protected String connection(String info) {
        System.out.println("마지막 접속단계");
        return info;
    }
}
```

Main Class

```
1  package TemplateMethod;
2
3  public class Main {
4      public static void main(String[] args){
5          GameConnectHelper helper = new DefaultGameConnectHelper();
6
7          helper.requestConnection( encodedInfo: "아이디 암호 등 접속 정보");
8      }
9  }
```

Summary

- 일정한 프로세스를 가진 요구사항을 템플릿 메소드로 구현

- 알고리즘의 구조를 메소드에 정의한다.
- 하위클래스에서 알고리즘 구조변경없이 재정의한다

Ex.

→알고리즘에 일정한 프로세스가있다

→ 변경 가능성이 큰 경우.